5.2. Microprocesoare PENTIUM

Prima generatie de Pentium, elaborata de Intel în mai 1993, a pastrat compatibiltatea cu microprocesoarele precedente, având posibiltatea de a executa doua instructiuni simultan, ceea ce îi confera o tehnologie superioara, comuna microprocesoarelor RlSC.

5.2.1 Arhitectura de baza

Arhitectura de baza (fig. 5.25) include urmatoarele structuri componente:

(1) Doua unitati de executie pentru operatii cu numere întregi (U si V) asimilate unor conducte *(pipes);* acestea lucreaza ca un ansamblu ce executa instructiunile microprocesorului, dar numai unitatea U poate executa setul complet de instructiuni.

Penlru cele doua unitati de executie se decodifica simultan doua instructiuni, iar executia lor se realizeaza tot simultan (cu conditia ca rezultalul celei de-a doua instructiuni sa nu depinda de rezultatul primei instructiuni). Aceasta conduce la o crestere a performantei cu circa 30%, Intel numind-o tehnologie superscalara.

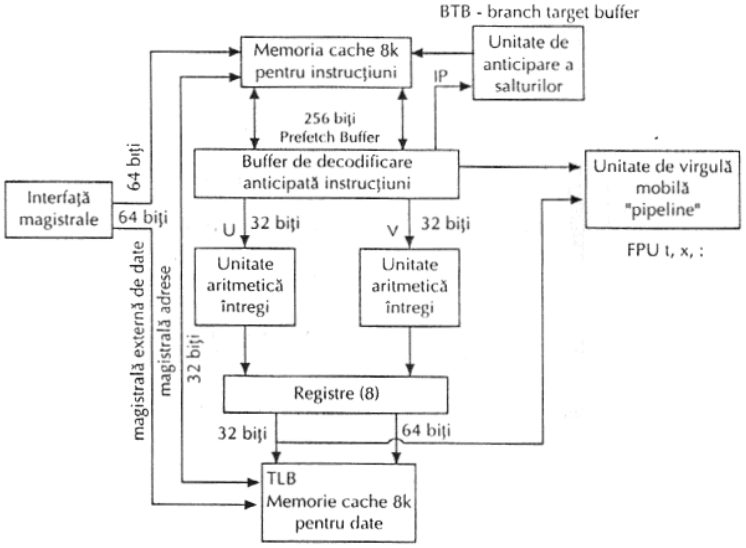


Fig. 5.25

De remarcat ca cele doua unitati lucreaza pe 32 de biti, ceea ce le confera compatibilitatea cu microprocesorul I80486.

(2) Unitatea de virgula mobila organizata lot sub forma unei linii de asamblare (conducta), ce contine unitati pentru executia hardware a adunarii, înmultirii si împartirii. Desi nu are viteza unui coprocesor, Pentium executa operatiile cu numere reale de doua ori mai rapid decat I80486.

(3) Memoria cache (nivel 1) este de doua ori mai mare decât cea a microprocesorului I80486 (16 K). fiind divizata în doua parti:

- 8 K pentru instructiuni - *code cache;*
- 8 K pentru date - *data cache.*

Circuilele de interfata integrate în cip descompun programul ce se executa în cuvinte de date si cuvinte de cod pe care le depun în memoriile cache corespunzatoare; accesul simultan la cele doua memorii cache permite introducerea datelor prin interfata magistralei concomitent cu citirile efectuate de unitatile de executie.

- in memoria cache pentru date, acestea pot fi modificate (rescrise) – "*write back";*
- in memoria cache pentru cod, informatiile nu pot fi modificale direct (pentru aceasta operatie este necesar un acces suplimentar la memoria DRAM) – "*write through".*

TLB *(Translation Lookaside Buffer)* translateaza adresa liniara în adresa fizica.

Se poate instala si cache secundar (nivel 2), de regula 16 K, având un timp de acces ≤ 15 ns si format din cipuri SRAM.

Magistrala de adrese este de 32 de biti, ceea ce ofera un spatiu de 2^{32} (4 G) memorie adresabila.
Magistrala externa de date este de 64 de biti, aceasta permitând transferul unul volum de date în/din microprocesor de doua ori mai mare decât pe magistralele de 32 de biti.

(4) Buffer de decodificare anticipata a instructiunilor.

Codul din memoria cache este testat pentru a sesiza din timp eventualele instructiuni de salt anterior incarcarii acestora în pipeline; decodificarea instructiunilor se realizeaza deci anticipat si ulterior sunt transmise unitatilor de executie. Transmiterea se realizeaza pe o magistrala de 256 de biti, dimensiunea mare a acesteia permitând aducerea secventelor de instructiuni cu o viteza mai mare chiar decât a capacitatii de procesare.

(5) Unitatea de anticipare a salturilor.

Aceasta unitate rezolva organizarea pipeline-ului, potrivit careia instructiunile sunt tratate într-o maniera strict secventiala, astfel ca atunci când apar instructiuni de salt, sa fie încarcata pe conducta secventa în ordinea executiei indicate de saltul specificat.

Etapele de executie simultana a doua instructiuni se desfasoara in 5 faze:
1. Încarcarea instructiunilor din memoria cache în bufferul de decodificare anlicipata.
2. Decodificarea unei instructiuni si calculul adresei.
3. Decodificarea urmatoarei instructiuni si calculul adresei.
4. Executia instructiunilor.
5. Depunerea rezultatelor executiei instructiunilor în memoria cache pentru date (rescrierea).

Se poate observa usor ca aceste faze ofera conditii de aparitie a unor stari conftictuale (de hazard) pe pipeline, astfel ca la un moment dat o instructiune aflata pe pipeline poate fi blocata într-o anumita faza, pâna când o instructiune încarcata anterior încheie executia unei faze si produce deblocarea. Pot aparea urmatoarele posibilitati de blocare:
- microprocesorul nu dispune de resursele necesare pentru a trata o anumita cornbinatie particulara de instructiuni;
- executia unei instructiuni de salt la o instructiune ce nu se afla în acel moment încarcata în memoria cache va bloca pipeline-ul pâna la încarcarea acestuia; anticiparea executiei urmatoarei instructiuni nu se va putea realiza în acest caz decât dupa executia instructiunii curente.

5.2.2. Pentium Pro, Pentium II, Celeron si Mendocino

1. **Prima generatie** a aparut în mai '93, lucra la frecventele de 60 si 66 MHz (P54C).
2. **A doua generatie** (martie '94) necesita schimbarea placii de baza:
- initial lucra la 90, 100 MHz; ulterior la 120, 133, 150, 166 si 200 MHz;
- placile de baza lucreaza la frecventa de 60, 66 MHz;
- indude APIC - *Advanced Programmable Interrupt Controller* (controler de întreruperi programabil avansat);
- dispune de interfata pentru procesor dual care suporta multiprelucrarea simetrica SMD - *Symetric Multiprocessing*, solicitata de sistemele de operare OS/2 si Windows NT.
3. **Pentium Pro** disponibil din 1996 la frecvente de 150, 166,180 s 200 MHz a introdus:
- 3 pipeline-uri pentru instructiuni interne;
- cacheul de date asociativ de 8 K cu patru cai de transfer si cache-ul de cod de 8 K cu doua cai pentru instructiuni primare;
- cache L2 (de nivel 2) de 256 K, 512 K integrat;
- executia dinamica a instructiunilor prin mecanismele de *branch prediction* si *speculative execution* permit executia instructiunilor în orice ordine (fig. 5.26).
4. **A treia generatie** (P55C). disponibila din ianuariel997 (socket 7 pe placa de baza), a încorporat tehnologia multimedia MMX *(Multimedia Execution)* în generatia a doua, având ca noutati:
- frecventa de 166,200,233 MHz;
- cache de cod de 16 K;
- adaugarea a 57 de instructiuni pentru functii audio, video si grafica.

Unitatea MMX include aplicatii multimedia si comunicatii care încorporeaza SIMD *(Single Instnuction Multiple Data)*, tehnica ce permite unei instructiuni sa execute anumite functii pe mai multe seturi de date.
5. **Pentium II** (P6) - Pentium Pro Klamath lansat în mai 1997 depaseste performantele unui Pentium 1a 200 MHz de circa 1,6 - 2 ori, lucrând 1a frecvente de 233, 266 si 300 MHz. Noutati incluse:
. cache-ul L1 dispune de 16 K pentru cod si 16 K pentru date;
- cache-ul L2 integrat de 512 K - 1 M;

- arhitectura DIB *(Dual Independent Bus*); magistrala duala independenta, una pentru cache-ul L2 si alta pentru memoria interna.

6. **Pendum II** (P6) - Pentium Pro Deschutes lucreaza Ia 300, 333 MHz.

Ambele variante accepta o arhitectura AGP *(Accelerated Graphic Port),* care permite conectarea unui subsistem grafic la selul de cipuri printr-o magis trala dedicata de mare viteza, ce degreveaza bus-ul PCI de transferul unui volum mare de date.

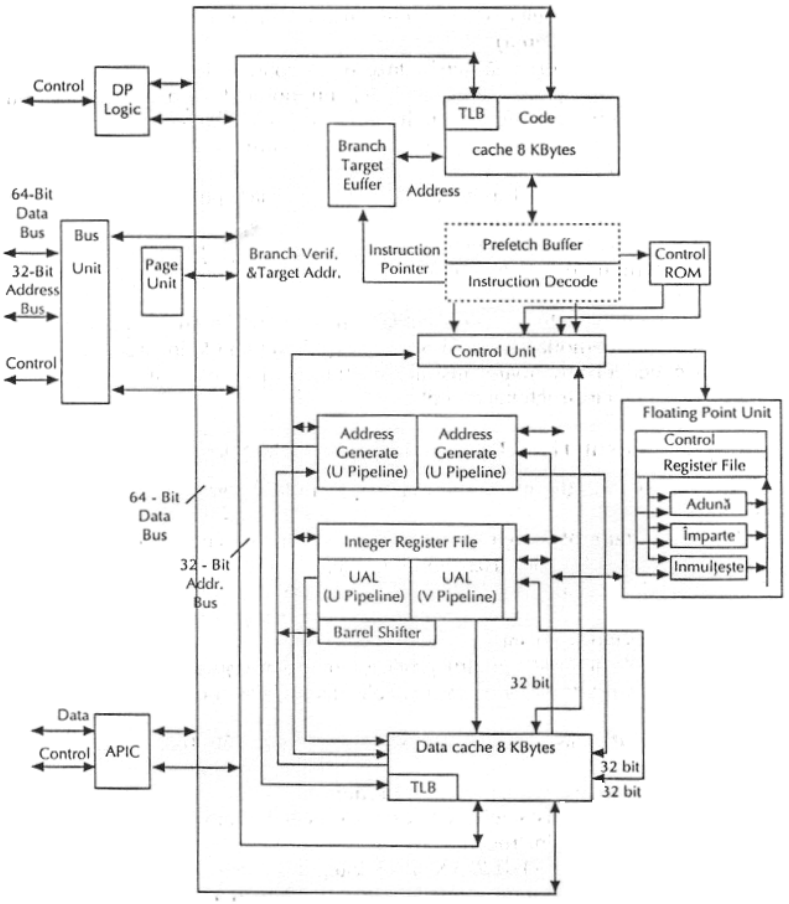


Figura 2.26. Arhitectura Pentium Pro

PCI asigura o rata de transfer de 132 M/s, iar AGP, la 66 MHz si 32 de biti, 533 M/s.

Memoria video pentru AGP este alocata dinamic, în functie de necesitati, din memoria sistemului si poate fi accesata rapid de controlerul grafic.

7. **Celeron** si **Mendocino**. Un alt reprezenlant al noii strategii Intel este procesorul Celeron. El a fost conceput pentru productia de masa, sub numele de cod "Covington", pentru a inlocui Pentium-ul în Basic PC, adica în calculatoarele din categoria de pret de sub 1.000 USD. Initial, anuntul firmei Inlel ca va opri productia procesoarelor Pentium a declansat un val de proteste pentru ca urmasul sau, Pentium II, fiind mai scump, nu era foarte potrivit pentru piata de larg consum. În industrie s-au înmultit producatorii care cereau sisteme Pentium rentabile pentru soclul 7, dar la momentul respectiv aceste procesoare nu mai erau fabricate decât de firmele concurente, AMD si Cyrix. Totusi, pentru a pastra atractiva si piata *low-end* (datorita numarului mare de produse ce pot fi vândute) s-a introdus în fabricatie Celeron-ul. În principal, Celeron este un Pentium II la care s-a renuntat la cache-ul secundar (L2 Cache). Un alt produs, dezvoltat sub numele de cod "Mendocino", a continuat linia Celeron, reintegrând cache-ul Level 2.

Renuntarea a cipurile integrale în Pentium II au permis producerea unui procesor mai simplu si mai ieftin, datorita faptului ca nu necesita placuta de siliciu care contine memoria cache si carcasa de deviere a caldurii.

Mecanismul sau este numit de Intel SEPP *(Single Edge Processor Package).* Pentru aerisire este folosit un ventilator care ocupa toata suprarata placutei si este fixat direct pe procesor. La fel ca si Pentium II, Celeron este

conceput pentru slotul 1.

Pentru Intel, "Covington" reprezinta începutul noii linii, de procesoare Celeron, cu acelasi nucleu ca întreaga familie Pentium II. Aceasta linie este conceputa din start pentru domeniul *low-end* si este cunoscuta si sub numele de Basic PC. Deoarece în cazul motherboard-urilor ieftine se mai poate renunta si la alte componente, s-a ajuns la definirea unui alt tip de *motherboard* micro ATX, putin mai mic decat board-urile ATX obisnuite care nu includ anumite facilitati, de care dispune 440 LX: suport dual-procesor, geslionarea a mai mult de 256 MB RAM, suport pentru memorie ECC si controlul a mai mult de trei sloturi PCI.

Pentru a sugera cumparatorilor ca este vorba despre produse rentabile din punct de vedere al pretului, Intel a renuntat la denumirea de Pentium II pentru Basic PC si a lansat numele de Celeron.

5.2.3. Pentium Xeon, Katmai si Willamette

XEON, urmatorul Pentium II al Intel, dispune de urmatoarele facilitati:

1) ca si Pentium II, Xeon are memorie cache de nivel 2 de 512 K, 1 M sau 2; nouI cartus slot 2 ce poate lucra cu cache L2 de pâna la 2 M, functioneaza la aceeasi frecventa de tact cu CPU;

2) doua *cipset-uri* cu suport multiprocesor, ceea ce ofera posibilitatea utilizarii mai eficiente pentru servere si statii de lucru echipate cu pâna la patru procesoare;

3) frecventa de tact 400 MHz în cazu1 cache-ului L2 de 512 K si 1M, respectiv 450 MHz pentru un cache L2 de 2 M;

4) sporirea sigurantei, prin adaugarea unor componente si caracteristici pentru administrare si monitorizare:

- senzor termic pentru urmarirea temperaturii cipului;
- verificarea si corectarea erorilor de date aparute pe parcursul transferului pe magistrala sistemului si magistralele memoriei cache L2;
- suport complet pentru ca doua procesoare sa realizeze aceleasi operatii cu aceleasi date, urmata de verificarea rezultatelor;
- o magistrala de gestiune a sislemului pentru urmarirea CPU printr-o interfata pentru doua noi componente de memorie ROM destinate partajarii informatiei cu software-ul si hardware-ul de gestiune a sistemului;

5) utilizeaza o noua platforma numita arhitectura de memorie server extinsa, care dispune de doua moduri de adresare a memoriei pe 36 de biti, o extensie a adresei de pagini pe 36 de biti si o extensie a dimensiunii de pagina pe 36 de biti, ceea ce permite accesarea si adresarea a pâna la 64 G de RAM. Pretul estimat initial era intre 7.600 si 9.400 USD pentru o statie de lucru echipata cu acest cip.

Intel apreciaza ca Xeon egaleaza si chiar depaseste performantele masinilor RISC ale firmelor Sun (Space), Compaq (Alpha), Hewlett-Packard (LC) sau Silicon Graphics (MIPS).

În martie 1998, firma SALIENT a lansat sistemul bazat pe Intel Penlium II la 350 MHz implementat pe piesa de baza SALIENT cu cipset BX, ce ruleaza la 500 MHz (cipset-LX 440 LX rula la 333 MHz), iar magistrala PCI lucreaza la 100 MHz.

Va fi descrisa în continuare succint microarhitectura familiei de procesoare P6 (Pentium Pro, Pentium II si III). În figura 5.27 este prezentata schema de baza a familiei de procesoare P6.

Componentele de baza sunt:

- subsistemul de memorie: memoria cache Ll si L2, unitatea de interfata cu magistrala *(Bus Interface Unit),* unitatea de interfata cu memoria (Memory *Interface Unit)* si bufferul de reordonare al accesel0r la memorie *(Memory Reorder Buffer);*
- unitatea de extragere/planificare - *Fetch/Decode Unit;* ,
- bufferul de reordonare a instructiunilor - *Reorder Buffer;*
- unitatea de planificare/executie a instructiunilor, formata dintr-o statie de rezervare centrala, doua unitati pentru întregi, o unitale pentru virgula mobila, doua unitati de generare a adreselor si doua unitati SIMD în virgula mobila (extensie SSE de la Pentium III),
- unitatea de validare a instructiunilor - *Retirement Unit.*

Subsistemul de memorie consta din memoria principala si cele 2 niveluri de memorie cache. Memoria principala este accesata prinlr-o magistrala de 64 biti, orientata pe tranzactii (fiecare acces este tratat separat). Memoriile cache de nivel 1 si 2 pot fi accesate la frecventa procesorului. Memoria cache de nivel 1 pentru instructiuni este organizata asociativ pe seturi cu 4 cai, iar cea pentru date pe 2 cai si suporta executarea unei operatii load si a uneia store pe ciclu.

Toate cererile unitatilor de executie catre memorie trec prin buffer-ul de reordonare *(Memory Reorder Buffer)* care functioneaza ca o unitate de planificare a instructiunilor *load/store.* Aici unele instructiuni *load/store* pot

fi reordonate pentru a mentine un flux continuu de instructiuni. Instructiunile *load* pot fi executate înaintea instructiunilor *store.* De asemenea, se pot executa si instrutiuni *load* speculative.

Unitatea de extragere si decodificare citeste instructiunile din memoria cache L1 si le decodifica într-o serie de instructiuni RISC denumite *micro-ops.* FIuxuI de *micro-ops* este îndreptat apoi catre ROB. Unitatea de extragere calculeaza si pointerul de instructiune pe baza informatiilor provenite din BTB (*Branch Target Buffer)* si a indicatiilor de predictie gresita a saltului, date de unitatea de lucru cu numere întregi. BTB este un *buffer* cu 512 intrari, ce contine informatii pentru predictia dinamica a salturilor. Datorlta faptului ca P6 are o banda de asamblare cu 12 niveluri, o predictie gresita implica o penalitate mare. De aceea, se utilizeaza o tehnica de predictie adaptiva cu 2 niveluri.

Unitatea de decodificare contine trei decodificatoare: doua pentru instructiuni simple (de obicei, aici se decodifica instructiunile ClSC într-o singura *micro-ops)* si unul pentru instructiuni complexe (raportul este o instructiune ClSC - 1...4 *micro-ops*). Exlsta si un secventiator de microoperatii, care ajuta la decodificarea instructunilor complexe. Decodificatorul poate sutine o rata de 6 *micro-ops/ciclu.*

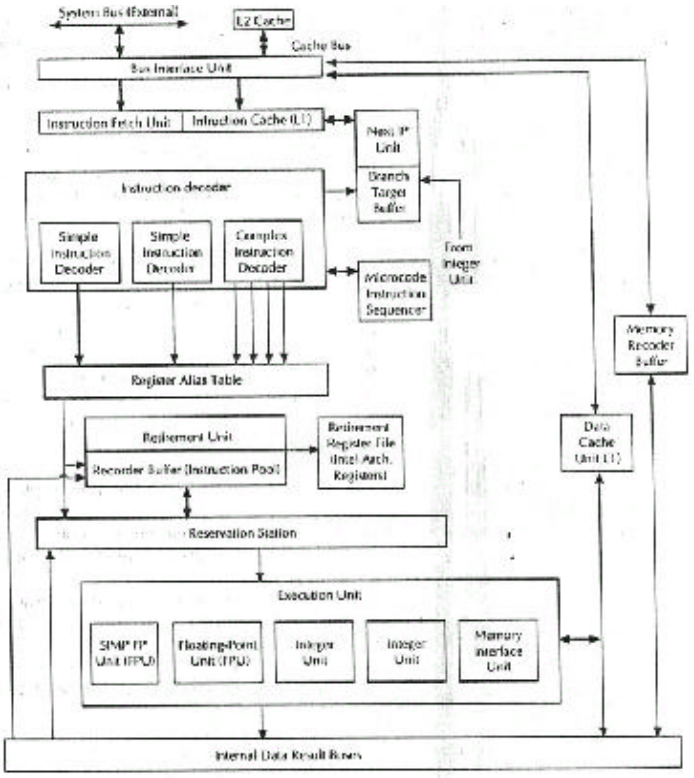


Fig. 5.27 Arhitectura familiei de procesoare P6

Pentru a rezolva problema dependentelor între instructiuni exista un set de 40 de registre fizice, care pot memora si valori întregi si în virgula mobila. Pentru a realiza redenumirea registrelor, se uti1izeaza un tabel de mapare *(Register alias Tobie),* unde referintele la registrele logice sunt transformate în referinte la registrele fizice. În etapa finala, toate instructiunile decodificate sunt trimise catre ROB.

Bufferul de reordonare (ROB) este organizat ca un masiv de memorie adresabila prin continut si are rolul de a memora atât instructiunile care asteapta executia, cât si pe cele care si-au terminat executia, dar nu au actualizat înca starea procesorului.

Unitatea de planificare are în componenta o statie de rezervare centrala, care este scanata mereu pentru a detecta instructiunile disponibile pentru planifacare la executie. Metoda de planificare este de tip *out-of-order:* se vor trimite pentru executie instructiunile care au toti operanzii disponibili, indiferent de ordinea în care ele apar în program. Când doua sau mai multe instructiuni de acelasi tip sunt disponibile pentru executie, ele vor fi selectate pe princlpiul FIFO.

Executia instructiuni1or este asigurata de doua unitati pentru numere întregi, doua pentru virgula mobila si o unitate pentru *load/store,* ceea ce permite ca 5 microinstructiuni sa poata fi planificate în fiecare ciclu.

Dintre cele doua unitati pentru întregi, una poate executa instructiunile de salt. Aceasta poate detecta predictiile gresite, pentru a senmala BTB sa restarteze banda de asamblare în mod corect.

Unitatea de interfata cu memoria poate executa atât o instructiune *load,* cât si una *store* în aceiasi ciclu, deoarece are doua unitati de calcul al adresei.

Unitatea de lucru cu numere în virgula mobila poate executa adunari (3 cicli), înmultiri (5 cicli), impartiri (18-38 clcli) si extragerea radacinii patrate (26-69 cicli).

Unitatea de validare a instructiunilor verifica permanent ROB, pentru a retrage instructiunile si a actualiza starea procesorului. Se pot extrage câte trei instructiuni/ciclu. La retragere, rezultatele sunt scrise în registrele destinate sau în memorie.

KATMAI continua seria de procesoare Pentium II, ce poarta un nume de cod care reprezinta numele extensiei setului de instructiuni a1 arhitecturii KNI: *Katmai New Instrudion Set.* Noile instrutiuni Katmai sustin aplicatiile multimedia, accelerând îndeosebi reprezentari grafice si video. Pentru a obtine într-adevar mai multa performanta prin intemediul acestor instructiuni, în beneficiul vitezei si a1 calitatii imaginii - ca si la MMX - este necesar software nou, care sa exploateze în mod efectiv noile instructiuni. Initial frecventa de tact a fost de 450 si 500 MHz, dar poate ajunge la 800 MHz.

Arhilectura Katmai a fost prevazuta cu o serie de caracteristici noi. Utililatea celor 70 de noi instructiuni si avantajele ce decurg din alte îmbunatatiri se combina.

O data cu MMX, Intel a introdus un procedeu care este denumit SIMD *(Single lnstruction/Multiple Data),* care are rolul de a prelucra simultan mai multe date similare, cum apar destul de des în aplicatii multimedia. MMX sunt doar instructiuni în virgula fixa, care nu accelereaza grafica 3D, unde se fac prelucrari în virgula mobi1a.

KNI extinde acest procedeu, care devine SIMD-FP (FP = *Floating Point),* incluzând si operatiile în virgula mobila. Pentru aceasta, KNI dispune de un set de opt registre independente, suplimentare, de câte 128 biti. În fiecare dintre aceste dintre aceste registre pot fi prelucrate simultan patru valori în virgula mobila, cu simpla precizie. În timp ce instructiunile MMX folosesc registrele FP ale coprocesorului, astfel încât nu este posibila utilizarea simultana a instructiunilor FP si a celor MMX, Katmai lucreaza cu registre suplimentare, care nu mai afecteaza registrele coprocesorului.

Pentru Windows 98 exista deja aceasta posibilitate. *Memory Streaming* este o alta caracteristica a arhitecturii Katmai. Programul comunica procesorului în prealabil care date preconizeaza ca vor fi încarcate. Katmai ofera multe optiuni, lasând la alegerea software-ului încarcarea datelor în toate cache-urile, doar în cache-ul L2 sau în nici un cache. Dupa prelucrare, datele pot fi scrise în cache sau direct în memorie.

Urmasul lui Katmai este Willamette, anuntat ca fiind ultimul din arhitectura XR6.

A doua generatie a procesorului Katmai este dezvoltata sub numele de cod «*Coppermine*». O data cu aceasta, Intel a introdus în locul tehnologiei de fabricatie a tranzistoarelor (0,25 rnicroni) structura de 0,18 microni, care permite, pe de o parte fabricarea de procesoare mai rapide, cu frecvente de tact care se situeaza la peste 500 MHz, iar pe de alta parte creeaza mai mult spatiu pentru componente suplimentare, care sunt integrate direct pe placuta de siliciu: includerea cache-ului L2, si aplicarea aceleiasi strategii ca la a doua generatie de Celeron (Mendocino), care contine cache-ul L2 pe procesor.

5.2.4. Intel Merced

Varianta P7 Merced, parteneriat cu Hewlett Packard, a vizat vizeaza atingerea urmatoarelor obiective:

- extinderea arhitecturii de 32 de biti la 64 de biti, pastrând însa compatibilitatea cu versiunile anterioare; nu necesita emulatoare de 16 si 32 de biti; ,
- adoptarea tehnologiei VLIW - *Very Long Instruction Word* (în opozitie cu proiectarea Intel pâna în prezent) care transfera sarcina optirnizarii sirului de instructiuni compilatorului, spre deosebire de P6 care efectueaza optimizarea în timpul executiei; deci Merced lucreaza cu filozofia rnicroprocesoarelor RlSC.

Un element esential al arhitecturii Merced este EPIC *(Explicitly Parallel Instruction Computing).* Numarul unitatilor *Integer* si *Floating Point* a fost marit. Sunt disponibile în acest sens câte 128 de registre.

Compilatorul este solicitat sa "vada" întregul program si sa "cunoasca" hardware-ul existent pentru prelucrare. Scopul este de a accelera ramificarile si buclele. Chiar si acum un procesor Pentium II poate executa anumite instructiuni de salt astfel încât economiseste timp fata de predecesoarele sale. Daca dupa aceea reiese ca anticiparea nu a fost corecta, calculul va fi reluat.

Datorita hardware-ului pe 64 de biti, al carui prim reprezentant a fost Merced, sunt disponibile destule resurse pentru a elimina ramificatiile. Compilatorul creeaza din acestea siruri de instructiuni independente, care pot fi prelucrate para1el în diferite unitati ale procesorului. În momentul în care drumul corect este stabilit, rezultatul gresit va fi pur si simplu eliminat.

IA-64 a fost dezvoltat în colaborare, de firrnele Intel si Hewlett Packard. Meced are o arhitectura complet diferita de procesoarele Pentium anterioare, însa poate executa si cod "x86" printr-un emulator hardware. Prima versiune a procesorului era preconizata a functiona la 600 MHz, însa versiunile ulterioare au ajuns la 900 MHz; cache-ul L2 integrat în procesor va fi de 2 MB sau 4 MB, iar viteza acestuia este de 100 MHz. Desi procesorul Merced a reprezentat una din optiuni, multi l-au evitat deoarece procesorul Intel Willamette este compatibil "x86".

5.2.5. Ramificatii si predictia ramificatiilor

Calculatoarele moderne lucreaza intens în banda de asamblare. Banda de asamblare are uneori 10 segmente sau chiar maj multe. Tehnica benzii de asamblare lucreaza cel mai bine pe cod liniar, astfel ca unitatea de citire poate extrage pur si simplu cuvinte consecutive din memorie si le poate trimite unitatii de decodificare mai înainte ca ele sa fie necesare.

Singura problema cu acest scenariu este aceea ca nu este tocmai realist. Programele nu sunt secvente liniare de cod. Ele sunt pline cu instructiuni de ramificatie. Sa consideram instructiunile din Fig. 5.28(a). O variabila i este comparata cu 0 (probabil cel mai obisnuit test în practica). În functie de rezultat, o alta variabila, k, va primi una din doua valori posibile.

```
if(i==0)                        CMPi,0  ;compara i cu 0
        k=1;                    BNE     ;salt la Else daca nu este egal
else                Then:       MOV j,1 ; încarca 1 în k
        k = 2                   BR Next ; salt neconditionat la Next
                    Else        MOV k,2 ; încarca 2 în k
                    Next
        (a)                             (b)
```

Figura 5.28. (a) Un fragment de program. (b) Translatarea sa într-un limbaj generic de asamblare.

O translatare posibila într-un limbaj de asamblare este aratata în figura 5.28(b). În functie de masina si de compilator, este probabil un cod mai mult sau mai putin asemanator cu cel din Fig. 5.28(b). Prima instructiune compara i cu 0. A doua transfera controlul la eticheta Else (începutul clauzei else) daca i nu este 0. A treia instructiune atribuie 1 lui k. A patra instructiune transfera controlul codului instructiunii urmatoare. Compilatorul a plasat acolo în mod convenabil o eticheta, Next deci exista un loc unde sa se sara. Cea de-a patra instructiune atribuie 2 lui k.

Ceea ce trebuie observat aici este ca din cele cinci instructiuni doua sunt de ramificatie. Mai mult una din acestea, BNE, este o ramificatje conditionata (saltul se face daca si numai daca o anumita conditie este îndeplinita, în acest caz daca cei doi operanzi din CMP precedent sunt egali). Cea mai lunga secventa liniara de cod este aici de doua instructiuni. În cosecinta, citirea instructiunilor la o rata mare pentru a alimenta banda de asamblare este foarte dificila.

La prima vedere se pare ca ramificatiile neconditionate ca instructiunea BR Next din Fig. 5.28(b) nu sunt o problema. Cu toate acestea, nu exista dubiu unde se sare. Dc ce sa nu poata unitatea de citire sa continue sa extraga instructiuni de la adresa tinta (locul unde se va sari)?

Problema consta în natura prelucrarii în banda de asamblare. De exemplu, decodificarea instructiunii are loc în al doilea segment. Astfel unitatea de citire trebuie sa decida unde sa extraga în continuare înainte sa cunoasca ce fel de instructiune tocmai a primit. De abia cu un ciclu mal târziu poate sa realizeze ca tocmai a primit o ramificatie neconditionata, dar începând deja sa citeasca instructiunea urmatoare ramificatiei neconditionate. Ca o consecinta, un numar substantial de masini în banda de asamblare (ca UltraSPARC II) au proprietatea ca instructiunea urmatoare unei ramificatii neconditionate este executata chiar daca logic nu ar trebui. Pozitia de dupa o ramificatie este numita pozitie de întârziere (delay slot). Pentium II §i masina utilizata în figura 5.28(b)] nu au aceasta proprietate, dar complexitatea interna de gestiune a acestei probleme este adesea enorma. Un compilator optimizant va încerca sa gaseasca unele instructiuni utile pe care sa le puna în pozitia de întârziere dar multe ori nu este nimic disponibil, asa ca este fortat sa insereze aici o instructiune NOP. În acest fel pastreaza programul corect, dar îl face mai mare si mai lent.

Pe cât sunt de suparatoare ramificatiile neconditionate, ramificatiile conditionate sunt si mai deranjante. Nu numai ca au de asemenea pozitii de întârziere, dar unitatea de citire nu are de unde sa citeasca, decât mult mai târziu în banda de asamblare. Masinile mai vechi cu banda de asamblare se blocau (stalled) doar pâna ce se cunostea daca

ramificatia avea loc sau nu. Blocarea pentru trei sau patru cicluni la fecare ramificatie conditionata, înrautateste dezastruos performanta în special daca 20% din instructiuni sunt ramificatii conditionate.

În consecinta, ceea cc fac majoritatea masinilor când întâlnesc o ramificatie conditionata, este sa prezica daca aceasta se va efectua sau nu. Au fost concepute diverse moduri de predictie. O varianta foarte simpla este urmatoarea: presupunem ca vor fi facute toate ramificatiile conditionate înapoi si ca nu vor fi facute cele inainte. Prima parte a presupunerii se justifica prin faptul ca ramificatii înapoi sunt foarte des întâlnite la sfârsitul unui ciclu (unei bucle). Majoritatea ciclurilor se executa de mai multe ori, deci mizând pe o ramificatie înapoi la începutul ciclului, în general, vom avea de câstigat.

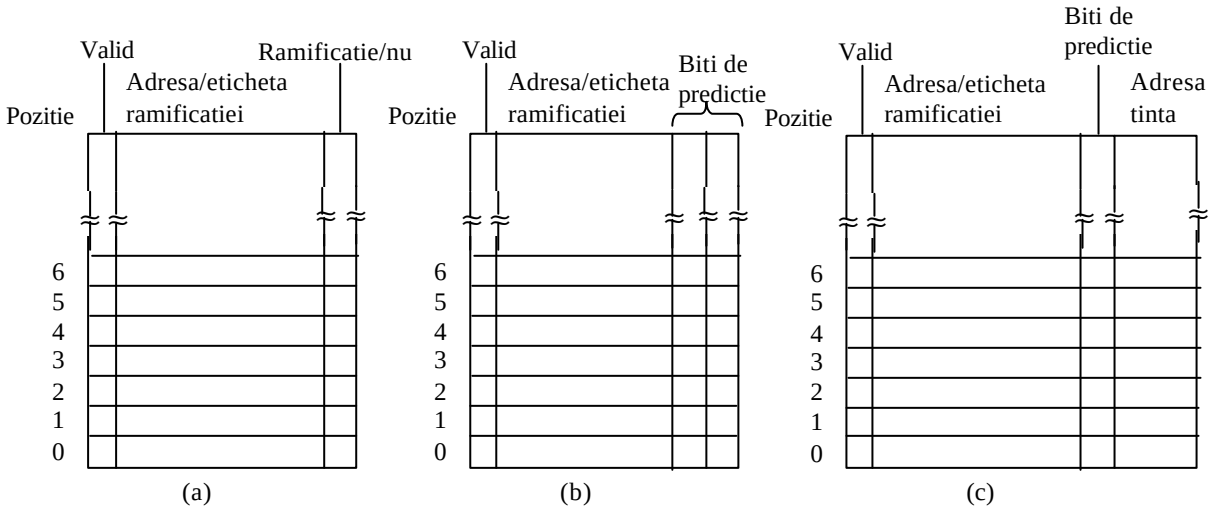
A doua parte este mai putin fondata. Se produc anumite ramificatii înainte când sunt detectate conditii de eroare în software (de exemplu, un fisier nu se poate deschide). Erorile sunt rare, asa ca majonitatea ramificatiilor asociate cu acestea nu sunt executate. Desigur, exista o multitudine de ramificatii înainte care nu sunt legate de gestiunea erorilor, asa ca rata de succes nu este chiar asa de buna ca la ramificatiile înapoi. Desi nu este fantastica, aceasta regula este oricum mai buna decât nimic.

Daca predictia unei ramificatii este corecta, nu este nimic special de facut. Executia continua la adresa tinta. Necazul apare când predictia unei raniificatii este gresita. Nu este dificil sa se estimeze unde sa se mearga si apoi sa se mearga acolo. Partea grea este sa se anuleze efectul instructiunilor care deja au fost executate si nu trebuiau.

Existii doua moduri de rezolvare. Prima cale este sa se permita executia instructiunilor citite dupa predictia unei ramificatii conditionate pâna când acestea încearca sa modifice starea masinii (de exemplu sa memoreze într un registru). În loc sa fie înscrisa în registru valoarea calculata este pusa într-un registru temporar (secret) si copiata în registrul real de abia dupa ce se stie ca predictia a fost corecta. Cea de a doua cale este sa se înregistreze valoarea oricarui registru înainte sa fie suprascris (de exemplu, într-un registru temporar secret), astfel ca masina poate fi adusa înapoi în starea pe care a avut-o înainte de predictia gresita. Ambele solutii sunt complexe si necesita o putere foarte mare de contabilizare pentru a functiona corect. Daca se atinge o a doua ramificatie conditionata înainte sa se cunoasca daca predictia primei a fost corecta, lucrurile se pot încurca serios.

Predictia dinamica a ramificatiilor

În mod clar, este foarte valoros ca predictiile s a fie exacte, caci acest lucru permite UCP sa lucreze la viteza maxima. În consecinta o mare parte a cercetarii curente se ocupa cu îmbunatatirea algonimilor de predictie a ramificatiilor (Driesen si Holzle 1998, Juan, 1998 Pan, 1992, Sechrest, 1996, Sprangle, 1997, Yeh si Pat,1991). O abordare este ca UCP sa mentina o tabela a istoriei (în special hardware), în care plaseaza ramificatiile conditionate pe masura ce acestea apar, astfel ca acestea pot fi cautate daca apar din nou. Versiunea cea mai simpla a acestei scheme este aratata în Fig. 5.29(a). Aici tabela istoriei contine o intrare pentru fiecare instructiune de ramificatie conditionata. Intrarea contine adresa instructiunii de ramificatie împreuna cu un bit care arata daca a fost efectuata ramificatia ultima data când a fost executata instructiunea. Utilizând aceasta schema, predictia consta în a spune pur si simplu ca ramificatia va urma aceeasi cale ca ultima data. Daca predictia este gresita, bitul din tabela istoriei este schimbat.



Figua 5.29. (a) Istorie de 1 bit a ramificatiilor (b). Istorie pe 2 biti a ramificatiilor.
(c) coresponclenta între adresa instructiunii de ramificatie si adresa tinta.

Exista mai multe moduri de organizare a tabelei istoriei. De fapt acestea sunt exact acelea moduri ca si cele utilizate pentru organizarea unui memorie intermediara. Considerând o masina cu instructiuni de 32 de biti cu aliniere la cuvânt, astfel încât cel mai putin semnificativi doi biti ai fiecarei adrese de memorie sunt 00. Cu o tabela a istoriei cu corespondenta directa continând 2 întrari, cel mai putin semnificativi n +2 biti ai unei instructiuni de ramificatie pot fi extrasi si deplasati dreapta cu 2 biti. Acest numar de n biti poate fi utilizat ca un index în tabela istoriei verificându-se daca adresa memorata acolo coincide cu adresa ramificatiei. La fel ca la memoria intermediara , nu este necesar sa se memoreze cei mai putin semnificativi n+2 biti, astfel ca ei pot fi omisi (adica numai bitii superiori de adresa - tag-ul - sunt memorati). Daca exista o coincidenta, bitul de predictie este utilizat pentru predictia ramificatiei. Daca este prezenta eticheta gresita sau intrarea este invalida, se produce o ratare, la fel ca la memoria intermediara. În acest caz, poate fi utilizata regula ramificatiei înainte/înapoi.

Daca tabela istoriei ramificatiilor are, sa zicem, 4096 de intrari, atunci ramificatiile de la adresele 0, 16384, 32768, ... vor fi în conflict, analog cu problema de la memoria intermediara. Aceeasi solutie este posibila: o intrare asociativa cu doua cai, patru cai sau n cai. La fel cu la memoria intermediara , cazul limita este o singura intrare asociativa de n cai, care necesita asociativitate completa.

În majoritatea situatiilor aceasta schema lucreaza bine pentru o dimensiune de tabela suficient de mare si suficienta asociativitate. Totusi, apare întotdeauna o problema sistematica. Când o bucla este în sfârsit încheiata predictia ramificatiei de la sfârsit va fi gresita, iar mai rau decât atât, predictia gresita va schimba bitul din tabela istoriei sa indice în viitor “non ramificatie”. Data viitoare când bucla este accesata, predictia ramificatiei de la sfârsitul primei iteratii va fi gresita. Daca bucla este interiorul altei bucle sau într-o procedura apelata frecvent, aceasta eroare poate sa apara foarte des.

Pentru a elimina aceasta predictie eronata putem sa-i dam intrarii tabelei o a doua sansa. Cu aceasta metoda, predictia este modificata numai dupa doua predictii incorecte consecutive. Aceasta abordare necesita doi biti de predictie în tabela istoriei, unul pentru ceca ce se presupune ca va face ramificatia si al doilea pentru ceea ce a facut ultima data, asa cum se arata în Fig. 5.29(b).

Un mod usor diferit de a privi acest algoritm este sa îl vedem ca o masina cu stari finite (FSM) cu patru stari, asa cum se arata în Fig. 5.30. Dupa o serie de predictii corecte “non ramificatie”, FSM va fi în starea 00 si va prezice “non ramificatie” data viitoare. Daca aceasta predictie este gresita ea se va muta în starea 10, dar prezicând de asemenea “non ramificatie” data viitoare . Numai daca a doua predictie este gresita se va muta în starea 11 si va prezice ramificatie de fiecare data. De fapt, bitul cel mai din stânga al starii este predictia si bitul cel mai din dreapta reprezinta ce a facut ramificatia ultima data. Aceasta solutie utilizeaza numai doi biti de istorie, dar sunt posibile proiecte care pastreaza cu patru sau opt biti de istorie.

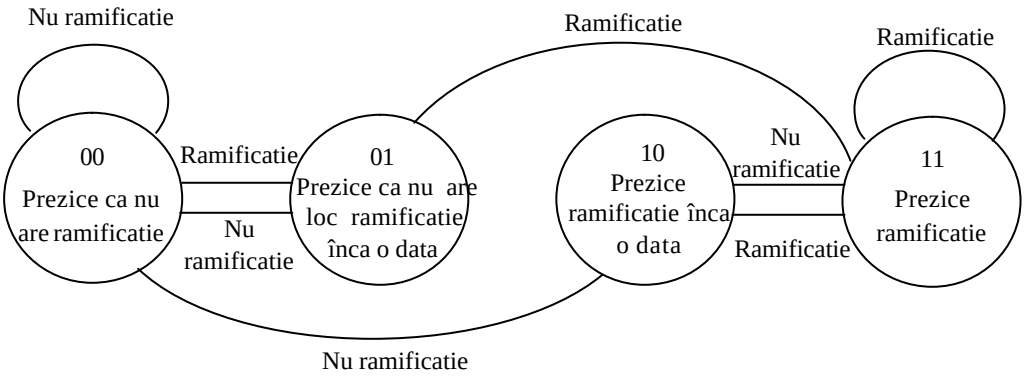


Figura 5.30. Masina cu stari finite pe 2 biti pentru predictia ramificatiilor**.**

De fapt, toate microprogramele noastre pot fi vazute ca FSM-uri, caci fiecare linie reprezinta o stare specifica în care se poate gasi masina, cu tranzitii bine definite spre un set finit de alte stari. FSM-unile sunt foarte larg utilizate în toate domeniile proiectarii hardware.

Pâna acum am presupus ca tinta fiecarei ramificatii conditionate era cunoscuta , în mod tipic fie ca o adresa explicita de ramificatie (continuta chiar în instructiune), fie ca o deplasare relativa la instructiunea curenta (adica, un

numar cu semn de adunat la contorul de program). Adeseori aceasta presupunere este valida , dar anumite instructiuni de ramificatie conditionata calculeaza adresa tinta prin operatii aritmetice asupra registrelor si apoi fac saltul acolo. Chiar daca FSM din Fig. 5.30 prezice corect ramificatiile de efectuat, o astlel de predictie nu este utila daca adresa tinta nu este cunoscuta . O cale de a aborda aceasta situatie este sa se memoreze adresa concreta din ramificatia efectuata ultima data în tabela istoriei, asa cum se arata în Fig. 5.29(c). În acest mod, daca tabela spune ca ultima data ramificatia de la adresa 516 a facut salt la adresa 4000, daca predictia acum este "ramificatie", presupunerea de lucru va fi salt din nou la 4000.

O abordare diferita a predictiei ramificatiilor este pastrarea informatiei referitoare la efectuarea ultimelor k ramificatii conditionate fara sa se tina seama ce instructiuni au fost (Pan, 1992). Acest numar de k biti, pastrat în registrul de deplasare al istoriei ramificatiilor (branch histoiy shift register), este comparat apoi în paralel cu toate intrarile unei tabele a istoriei cu o cheie de k biti si, daca este gasit, se utilizeaza predictia gasita acolo. Oarecum surprinzator, aceasta tehnica lucreaza destul de bine.

Predictia statica a ramificatiilor

Toate tehnicile de predictie a ramificatiilor discutate pâna acum sunt dinamice, adica generate în timpul rularii programului. Ele se adapteaza de asemenea la comportarea curenta a programului, ceea ce este bine. Partea neplacuta este ca necesita hardware special si costisitor si o mare complexitate a cipului.

Un alt mod de predictie se bazeaza pe ajutorul compilatorului. Când compilatorul vede o instructiune ca:

for (i=0; i<1000000; i++) {....}

el stie foarte bine ca ramificatia de la sfârsitul buclei se va efectua aproape tot timpul. Daca ar exista o cale de a comunica hardware-ului acest lucru, s-ar putea economisi un mare efort.

Desi aceasta este o modificare arhitecturala (nu doar o solutie de implementare), anumite masini, ca UltraSPARC II, au un al doilea set de instructiuni de ramificatie conditionata, în plus fata de cele obisnuite (care sunt necesare pentru compatibilitate cu sistemele mai vechi). Cele mai noi contin un bit în care compilatorul poate specifica daca presupune ca ramificatia va fi executata (sau nu). Când una dintre acestea este întâlnita, unitatea de fetch face exact ce i s-a spus. Mai mult, nu e nevoie sa se piarda spatiu pretios în tabela de istorie a ramificatiilor pentru aceste instructiuni, reducând astfel conflictele de acolo.

În final, ultima tehnica de predictie a ramificatiilor este bazata pe realizarea profi lului (profiling) (Fisher si Freudenberger, 1992). Aceasta este de asemenea o tehnica statica , dar în loc de a lasa compilatorul sa estimeze care ramificatii vor fi efectuate si care nu, programul este de fapt executat (de obicei pe un simulator), si se captureaza comportarea ramificatiilor. Aceasta informatie este introdusa în compilator, care utilizeaza apoi instructiunile speciale de ramificatie conditionta pentru a spune hardware-ului ce sa faca.